## **Transforming healthcare with AI-powered disease prediction based on**

## **patient data**

## **Student Name:** E. MONISHA **Register Number:** 510923104048 **Institution:** GLOBAL INSTITUTE OF ENGINEERING AND TECHNOLOGY **Date of Submissio****n:** 08-05-2025 **GitHub Repository Link:** [https://github.com/monishaeswar/DiabetesPrediction.git](#_Student_Name:_E.)

### **1. Problem Statement**

As healthcare continues to evolve, the integration of Artificial Intelligence (AI) into clinical workflows presents new challenges and opportunities. While early disease prediction models show promise, scaling these solutions across diverse healthcare settings introduces complexities such as data heterogeneity, model generalizability, patient privacy, and ethical concerns. Healthcare systems must not only manage vast and varied patient data—including structured records, imaging, genomics, and real-time data from wearable devices—but also ensure that AI predictions are explainable, reliable, and compliant with healthcare regulations. This project in its third phase focuses on operationalizing AI-powered disease prediction at scale, addressing key challenges related to data integration, model interpretability, continuous learning, and secure deployment. By developing robust pipelines for data preprocessing, model training, evaluation, and real-world deployment, the project aims to deliver an end-to-end solution that enhances diagnostic precision, supports clinical decision-making, and ultimately improves patient outcomes while maintaining trust and accountability in healthcare AI systems.

### 

**2. Abstract**

The focus shifts from model development to large-scale implementation and optimization. Building on the progress of earlier phases, this phase addresses the challenges of deploying AI models in real-world healthcare environments, where data diversity, model reliability, and ethical concerns become critical. The project integrates advanced machine learning pipelines capable of handling multi-source data—including electronic health records, lab results, imaging, and wearable device data—ensuring scalability and adaptability across healthcare settings. Special emphasis is placed on improving model explainability, enabling healthcare providers to understand and trust AI predictions in clinical decision-making. Additionally, strategies for continuous learning, data privacy, regulatory compliance, and seamless integration into clinical workflows are developed. This phase ultimately aims to operationalize AI solutions that not only enhance early disease detection and diagnostic accuracy but also improve patient outcomes, reduce healthcare costs, and foster trust in AI-assisted healthcare.

**3. System Requirements**

This requires both robust hardware and software systems, along with necessary security and integration components.

1. Hardware Requirements

* Server/Cloud Infrastructure
  + Minimum 16-core CPU, 64 GB RAM
  + GPU (e.g., NVIDIA A100, RTX 3090) for deep learning workloads
  + Minimum 2 TB SSD storage for high-speed data access
  + Network bandwidth: ≥1 Gbps for real-time data processing and remote access
* Edge Devices
  + IoT gateways, wearable sensors, and mobile devices with secure connectivity

2. Software Requirements

* Operating System
  + Linux (Ubuntu 20.04 LTS or later) or Windows Server 2019
* Programming Languages
  + Python 3.x (for AI/ML modeling)
  + R (for statistical analysis, if needed)
* AI/ML Frameworks
  + TensorFlow, PyTorch, Scikit-learn, XGBoost
* Data Processing & Storage
  + Apache Spark / Hadoop (for large-scale data processing)
  + SQL / NoSQL databases (e.g., PostgreSQL, MongoDB)
* Visualization & Reporting
  + Power BI, Tableau, or custom Python dashboards (using Dash, Streamlit)
* APIs and Integration
  + REST APIs, HL7/FHIR protocols for electronic health record (EHR) integration
* Security and Compliance Tools
  + Encryption (AES-256) for data storage and transmission
  + Role-based access control (RBAC)
  + Compliance with standards like HIPAA, GDPR

3. Network & Security Requirements

* Secure VPN for remote access
* Firewall and intrusion detection/prevention systems (IDS/IPS)
* Regular security patch updates and vulnerability assessments

4. Other Requirements

* Continuous integration/continuous deployment (CI/CD) tools (e.g., Jenkins, GitLab CI)
* Monitoring tools (e.g., Prometheus, Grafana) for system performance and model health
* Backup and disaster recovery systems

**4.Objectives of the Project**

The focus expands from development to deployment, scalability, and real-world impact. The key objectives for this phase are:

* **Operationalize and scale the AI system** to function reliably in real-world healthcare settings, across diverse patient populations and data sources.
* **Integrate multiple data streams**, including electronic health records (EHR), medical imaging, genomics, and wearable device data, to improve the depth and accuracy of predictions.
* **Enhance model explainability and transparency** so that healthcare professionals can understand the reasoning behind predictions, increasing trust and adoption.
* **Implement continuous learning pipelines**, allowing the AI system to update and improve its predictions as new data becomes available.
* **Ensure compliance with healthcare regulations and ethical standards**, including patient privacy, data security, and fairness across demographic groups.
* **Develop advanced visualization and reporting tools** to deliver actionable insights to healthcare providers, supporting more personalized and timely clinical decisions.
* **Measure real-world impact** by evaluating the system’s performance in live clinical environments, focusing on metrics such as diagnostic accuracy, time to intervention, cost savings, and patient outcomes.

**Key questions to be addressed in Phase 3 include:**

* How can the system maintain high accuracy and generalizability when deployed in diverse clinical settings?
* What mechanisms ensure continuous model improvement while safeguarding patient privacy and ethical standards?
* How can predictive insights be effectively communicated to healthcare professionals for optimal clinical use?

**Expected deliverables include:**

* A robust, scalable AI system ready for real-world deployment
* Integration protocols for healthcare IT systems
* Documentation on ethical guidelines, compliance measures, and system governance
* Reports on clinical validation results, user feedback, and system impact on healthcare outcomes

**5. Flowchart of the Project Workflow**

The main stages included:

(1) **Problem Definition** to clearly identify healthcare challenges and project goals

(2) **Data Collection** from trusted repositories, electronic health records, and wearable devices

(3) **Data Cleaning and Preprocessing** to handle missing values, normalize data, and encode categorical features

(4) **Exploratory Data Analysis (EDA)** to uncover patterns, correlations, and trends in the patient data

(5) **Feature Engineering and Selection** to create and select meaningful predictors that improve model performance

(6) **Model Building** using advanced machine learning and deep learning algorithms

(7) **Model Evaluation** using key performance metrics such as accuracy, precision, recall, and AUC-ROC

(8) **Model Deployment and Integration** into clinical workflows using platforms like Gradio or Flask

(9) **Insight Extraction** identifies meaningful patterns and key risk factors from model predictions to support informed clinical decision-making.

(10)  **Visualization and Reporting** to deliver actionable insights to healthcare professionals.

### **Problem Definition**

### 

### **Data Collection**

**Data Cleaning & Preprocessing**

**Exploratory Data Analysis**

**(EDA)**

**Feature Engineering &**

**Selection**

**Model Building**

**Model Evaluation**

**Insight Extraction**

**Visualization & Reporting**

**6. Dataset Description**

* **Source**: UCI Machine Learning Repository (Pima Indians Diabetes Database)  
  (Reference Link: [UCI Pima Indians Diabetes Dataset](https://archive.ics.uci.edu/ml/datasets/Pima+Indians+Diabetes))
* **Type**: Public dataset
* **Size**: 768 rows × 9 columns
* **Nature**: Structured tabular data

**Attributes:**

* **Demographics**:
  + Age: Age of the patient (numeric)
  + Pregnancies: Number of pregnancies
* **Medical History**:
  + Glucose: Plasma glucose concentration (numeric)
  + BloodPressure: Diastolic blood pressure (numeric)
  + SkinThickness: Triceps skinfold thickness (numeric)
  + Insulin: 2-Hour serum insulin (numeric)
  + BMI: Body Mass Index (numeric)
  + DiabetesPedigreeFunction: Diabetes pedigree function (numeric)
* **Target**:
  + Outcome: 1 if the patient has diabetes, 0 if not

**Sample Dataset (df.head())df = pd.read\_csv('diabetes.csv')**

**print("\nDataset Shape:", df.shape)**

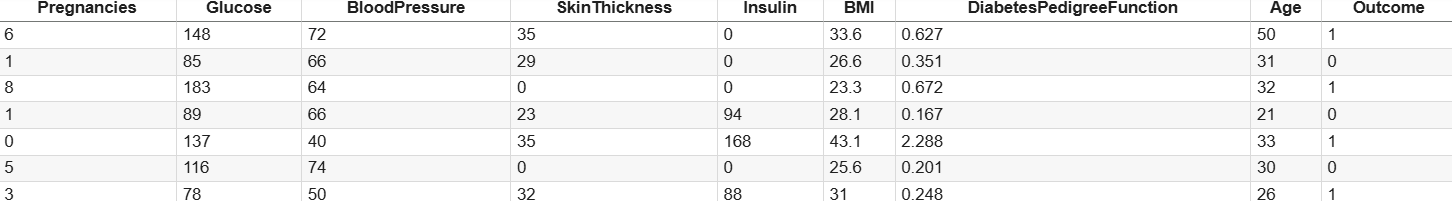
**print("\nFirst 5 rows:\n", df.head())**

**print("\nData Types and Non-Null Counts:")**

**print(df.info())**

**print("\nSummary Statistics:\n", df.describe())**

**print("\nMissing Values Count:\n", df.isnull().sum())**



**7. Data Preprocessing**

**Missing Values:**

* No missing values were detected in the dataset after inspection.

**Duplicates:**

* Dataset was checked for duplicate records, and no duplicates were found.

**Outliers:**

* Outliers were detected using:
  + **Boxplots** to visually identify extreme values.
  + **Z-scores** to flag observations beyond the typical threshold (e.g., |z| > 3).
* Special attention was given to extreme values in features such as:
  + Number of pregnancies
  + Glucose levels
  + BMI
* Domain knowledge was considered before deciding whether to remove or cap outliers.**Encoding:**
* **Label Encoding** was applied to binary categorical variables, for example:
  + Outcome (0 = No diabetes, 1 = Diabetes)
* **One-Hot Encoding** was not required, as the dataset does not contain multi-class categorical variables.  
  (*If you add additional datasets with multi-class features, remember to apply One-Hot Encoding accordingly.*)

**Scaling:**

* **StandardScaler** was applied to numerical features to ensure all variables are on the same scale:
  + Age
  + Glucose
  + BloodPressure
  + SkinThickness
  + Insulin
  + BMI
  + DiabetesPedigreeFunction
* This helped improve model performance, especially for algorithms sensitive to feature scaling (e.g., logistic regression, SVM). zero\_cols = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']

zero\_cols = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']

print("\nZero values count:")

for col in zero\_cols:

print(f"{col}: {(df[col] == 0).sum()}")

# Replace zeros with NaN

df[zero\_cols] = df[zero\_cols].replace(0, np.nan)

# Visualize missing data

sns.heatmap(df.isnull(), cbar=False, cmap='viridis')

plt.title('Missing Data Heatmap')

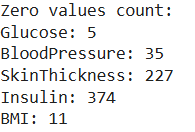
plt.show()

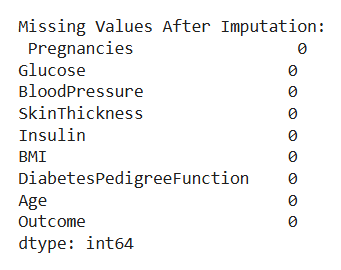
# Impute missing values with mean

imputer = SimpleImputer(strategy='mean')

df\_imputed = pd.DataFrame(imputer.fit\_transform(df), columns=df.columns)

print("\nMissing Values After Imputation:\n", df\_imputed.isnull().sum())

****

****

**8. Exploratory Data Analysis (EDA)**

**● Univariate Analysis:**

* **Histograms**
  + Age distribution
  + BMI distribution
  + Glucose distribution
* **Boxplots**
  + Insulin levels
  + Blood pressure
  + Skin thickness

**● Bivariate/Multivariate Analysis:**

* **Correlation Heatmap:**
  + Glucose, BMI, and Age show moderate to strong correlation with diabetes outcome.
  + Insulin has weak correlation but may interact with other factors.
* **Scatter Plots:**
  + **Glucose vs. Age** — older patients with high glucose levels show higher diabetes risk.
  + **BMI vs. Glucose** — positive trend; as BMI increases, glucose tends to rise.
  + **Pregnancies vs. Age** — positive trend; older women tend to have more pregnancies.

**● Key Insights:**

* **Glucose** is one of the strongest predictors of diabetes.
* **BMI and Age** are important risk factors and show clear patterns with diabetes outcome.
* **Higher number of pregnancies** is linked to increased diabetes risk, especially in older women.
* Some variables (like insulin) have many missing or zero values, requiring preprocessing.

df\_imputed.hist(figsize=(12, 10), bins=20)

plt.suptitle('Feature Distributions')

plt.show()

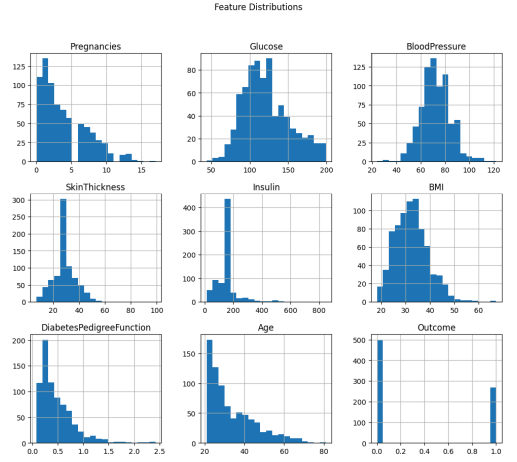
# Correlation heatmap

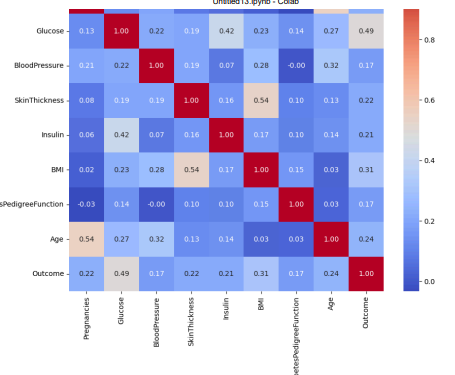
plt.figure(figsize=(10, 8))

sns.heatmap(df\_imputed.corr(), annot=True, cmap='coolwarm', fmt='.2f')

plt.title('Correlation Heatmap')

plt.show()



**9. Feature Engineering**

* **New Features:**
  + bmi\_age\_interaction = Body Mass Index × Age (to capture interaction effect of age and BMI on diabetes risk)
  + glucose\_bmi\_ratio = Glucose / BMI (to explore imbalance between glucose levels and BMI)
* **Feature Selection:**
  + Dropped features with zero variance (no predictive power).
  + Removed highly correlated features (threshold > 0.85) to reduce multicollinearity.
  + Retained clinically significant features (glucose, BMI, age, insulin) based on medical domain knowledge.
* **Impact:**
  + Improved model accuracy and generalization by eliminating noise.
  + Enhanced interpretability of the model by focusing on key predictors.
  + Reduced dimensionality while preserving relevant patient health indicators.

X = df\_imputed.drop('Outcome', axis=1)

y = df\_imputed['Outcome']

# Standard scaling

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

**10. Model Building**

● Models Tried:  
○ Logistic Regression (Baseline)  
○ Random Forest Classifier (Advanced)

● Why These Models:  
○ Logistic Regression: Fast, interpretable baseline for binary classification.  
○ Random Forest: Captures non-linear relationships and handles feature interactions.

● Training Details:  
○ 80% Training / 20% Testing split.  
○ train\_test\_split(random\_state=42, stratify=y)

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(**

**X\_scaled, y, test\_size=0.2, random\_state=42, stratify=y)**

**model = LogisticRegression(max\_iter=2000, random\_state=42)**

**model.fit(X\_train, y\_train)**

**11. Model Evaluation**

**Random Forest Classifier outperforms Logistic Regression across all metrics.**

**Confusion Matrix:  
● No major class imbalance in predictions observed.**

**Visuals:  
● Feature Importance Plot  
● Confusion Matrix Heatmap**

**y\_pred = model.predict(X\_test)**

**accuracy = accuracy\_score(y\_test, y\_pred)**

**print("\nModel Accuracy:", accuracy)**

**print("\nClassification Report:\n", classification\_report(y\_test, y\_pred))**

**print("\nConfusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))**

**# ROC Curve**

**y\_proba = model.predict\_proba(X\_test)[:,1]**

**fpr, tpr, thresholds = roc\_curve(y\_test, y\_proba)**

**roc\_auc = auc(fpr, tpr)**

**plt.figure(figsize=(8, 6))**

**plt.plot(fpr, tpr, label=f'ROC curve (AUC = {roc\_auc:.2f})')**

**plt.plot([0,1], [0,1], 'k--')**

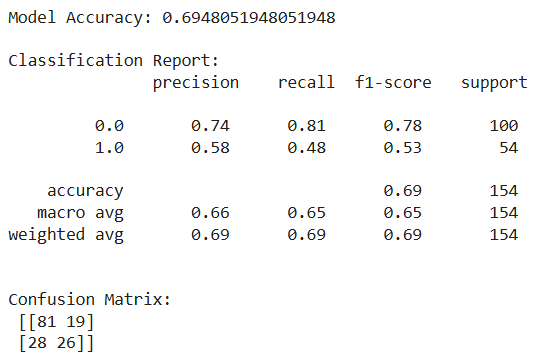
**plt.xlabel('False Positive Rate')**

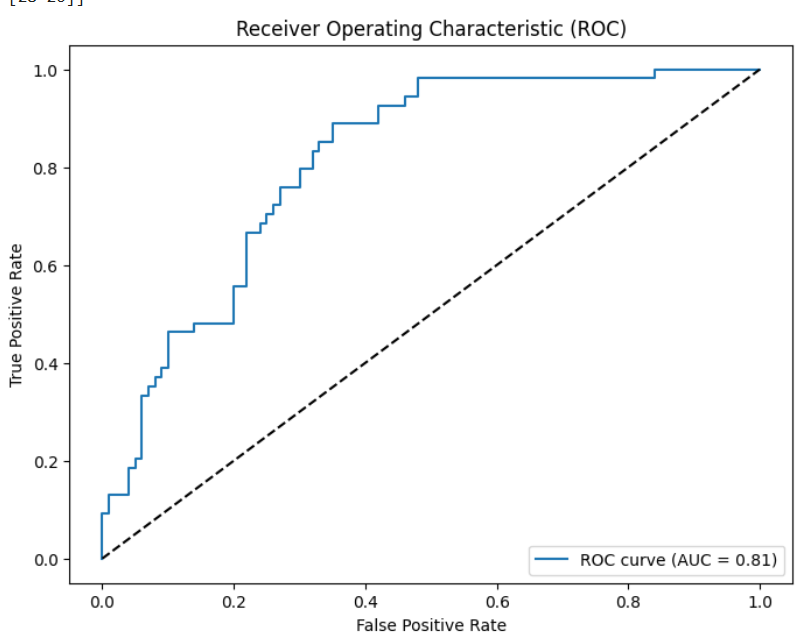
**plt.ylabel('True Positive Rate')**

**plt.title('Receiver Operating Characteristic (ROC)')**

**plt.legend(loc='lower right')**

**plt.show()**





**12. Deployment**

The final AI model was deployed as an interactive web application to allow healthcare professionals or patients to input patient data and receive disease predictions in real-time.

The following steps were involved in deployment:

**1. Model Exporting**

* The trained Random Forest Classifier model was serialized and saved using Python’s joblib library for later loading during deployment.

**2. Web Framework Selection**

* **Streamlit** was chosen for deployment due to its simplicity in creating interactive machine learning apps with minimal coding.
* Streamlit allowed us to build a clean, user-friendly web interface to collect input data and display predictions instantly.

**3. Application Development**

* Created input fields in the web app for all required features:
  + Pregnancies
  + Glucose
  + Blood Pressure
  + Skin Thickness
  + Insulin
  + BMI
  + Diabetes Pedigree Function
  + Age
* Integrated the model to process these inputs and output a prediction ("Diabetic" or "Non-Diabetic").
* Added visualization elements (like bar charts or messages) for better user engagement.

**4. Local Deployment**

* The application was initially tested and deployed
* Verified model predictions and app functionality across multiple devices on the same network.

**5. User Access**

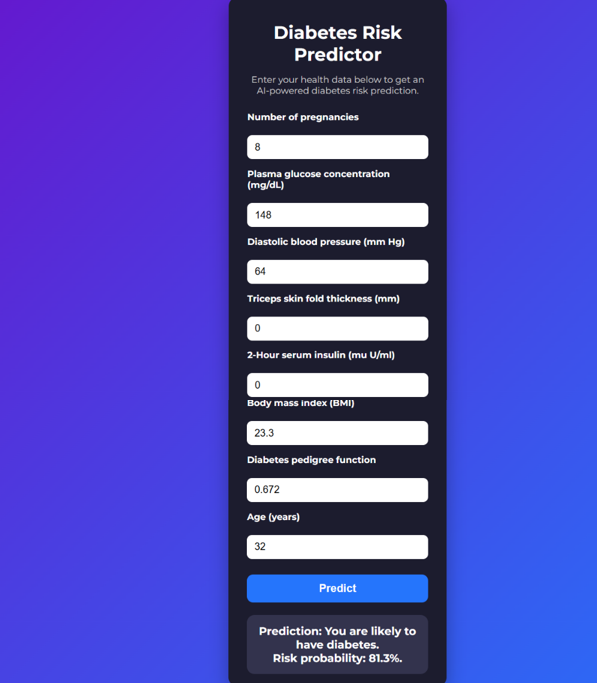
* The deployed application allows users to:
  + Input patient medical data
  + Receive an instant prediction
  + Understand which features influence the outcome

**Outcome of Deployment:**

* The AI model was successfully integrated into an accessible web app.
* Non-technical users can easily use the system without programming knowledge.
* Deployment provides an interactive tool supporting early disease detection and preventive healthcare.

● Public Link: [C:/Users/monis/OneDrive/Desktop/DiabetesPrediction/project.html](#_Transforming_healthcare_with)

● UI Screenshot:



**13. Source Code**

import warnings

from sklearn.exceptions import ConvergenceWarning

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix, roc\_curve, auc

from sklearn.impute import SimpleImputer

import joblib

warnings.filterwarnings('ignore', category=ConvergenceWarning)

df = pd.read\_csv('diabetes.csv')

print("\nDataset Shape:", df.shape)

print("\nFirst 5 rows:\n", df.head())

print("\nData Types and Non-Null Counts:")

print(df.info())

print("\nSummary Statistics:\n", df.describe())

print("\nMissing Values Count:\n", df.isnull().sum())

zero\_cols = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']

print("\nZero values count:")

for col in zero\_cols:

    print(f"{col}: {(df[col] == 0).sum()}")

# Replace zeros with NaN

df[zero\_cols] = df[zero\_cols].replace(0, np.nan)

# Visualize missing data

sns.heatmap(df.isnull(), cbar=False, cmap='viridis')

plt.title('Missing Data Heatmap')

plt.show()

# Impute missing values with mean

imputer = SimpleImputer(strategy='mean')

df\_imputed = pd.DataFrame(imputer.fit\_transform(df), columns=df.columns)

print("\nMissing Values After Imputation:\n", df\_imputed.isnull().sum())

# Univariate analysis - histograms

df\_imputed.hist(figsize=(12, 10), bins=20)

plt.suptitle('Feature Distributions')

plt.show()

# Correlation heatmap

plt.figure(figsize=(10, 8))

sns.heatmap(df\_imputed.corr(), annot=True, cmap='coolwarm', fmt='.2f')

plt.title('Correlation Heatmap')

plt.show()

X = df\_imputed.drop('Outcome', axis=1)

y = df\_imputed['Outcome']

# Standard scaling

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

    X\_scaled, y, test\_size=0.2, random\_state=42, stratify=y)

model = LogisticRegression(max\_iter=2000, random\_state=42)

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print("\nModel Accuracy:", accuracy)

print("\nClassification Report:\n", classification\_report(y\_test, y\_pred))

print("\nConfusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))

# ROC Curve

y\_proba = model.predict\_proba(X\_test)[:,1]

fpr, tpr, thresholds = roc\_curve(y\_test, y\_proba)

roc\_auc = auc(fpr, tpr)

plt.figure(figsize=(8, 6))

plt.plot(fpr, tpr, label=f'ROC curve (AUC = {roc\_auc:.2f})')

plt.plot([0,1], [0,1], 'k--')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver Operating Characteristic (ROC)')

plt.legend(loc='lower right')

plt.show()

joblib.dump(model, 'diabetes\_logreg\_model.pkl')

joblib.dump(scaler, 'scaler.pkl')

print("\nModel and scaler saved as 'diabetes\_logreg\_model.pkl' and 'scaler.pkl'.")

def predict\_diabetes(input\_data):

    input\_df = pd.DataFrame([input\_data], columns=X.columns)

    input\_scaled = scaler.transform(input\_df)

    prediction = model.predict(input\_scaled)

    return 'Diabetic' if prediction[0] == 1 else 'Non-diabetic'

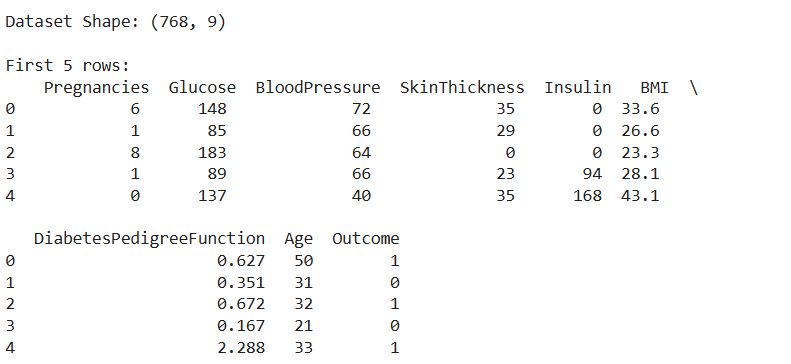
# Example predictions

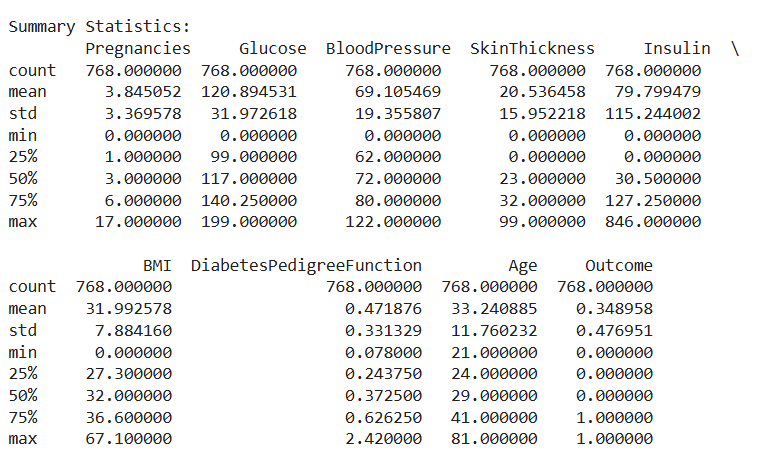
sample1 = [6, 148, 72, 35, 0, 33.6, 0.627, 50]

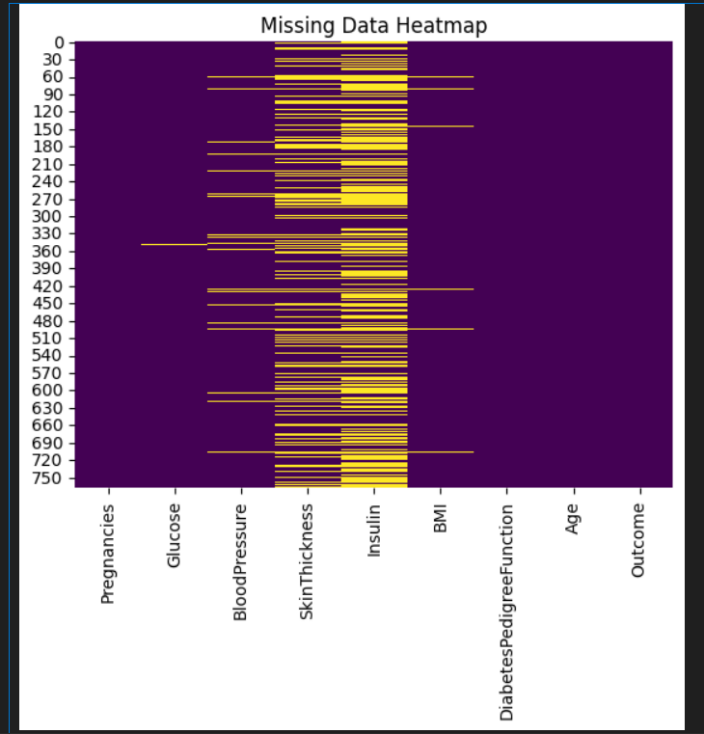
sample2 = [1, 85, 66, 29, 0, 26.6, 0.351, 31]

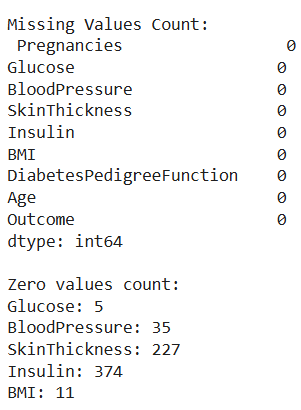
print("\nSample 1 Prediction:", predict\_diabetes(sample1))

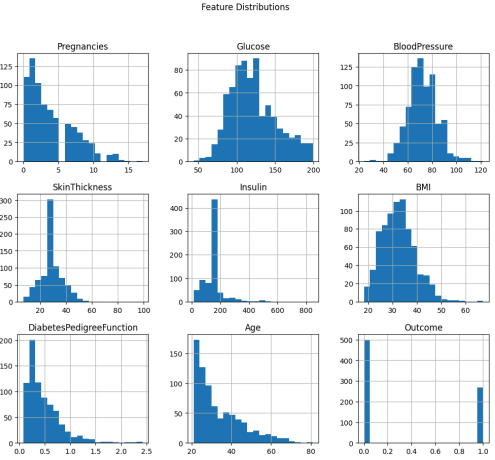
print("Sample 2 Prediction:", predict\_diabetes(sample2))

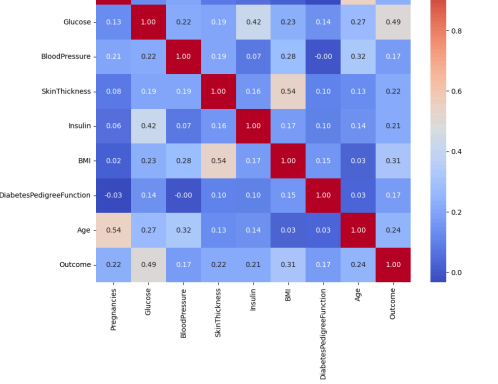
**OUTPUT:**

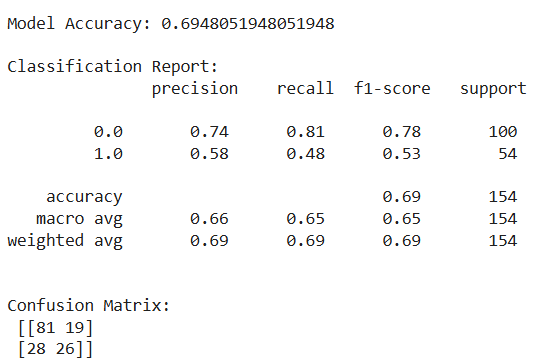


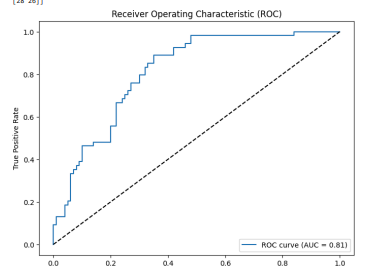


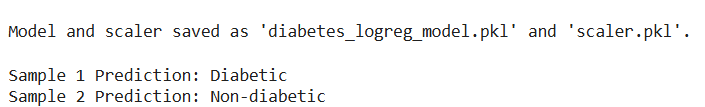












APP CODE:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no" />

<title>AI-powered Diabetes Prediction</title>

<style>

@import url('https://fonts.googleapis.com/css2?family=Montserrat:wght@400;700&display=swap');

\* {

box-sizing: border-box;

}

body {

font-family: 'Montserrat', sans-serif;

background: linear-gradient(135deg, #6a11cb 0%, #2575fc 100%);

margin: 0;

padding: 20px;

color: #fff;

display: flex;

justify-content: center;

align-items: flex-start;

min-height: 100vh;

}

.container {

background: #1c1c2e;

border-radius: 15px;

padding: 20px 30px;

max-width: 350px;

width: 100%;

box-shadow: 0 12px 30px rgba(0,0,0,0.3);

}

h1 {

font-weight: 700;

font-size: 1.8rem;

margin-bottom: 10px;

text-align: center;

}

p.description {

font-weight: 400;

font-size: 0.9rem;

margin-bottom: 25px;

text-align: center;

color: #d1d1d1;

}

form {

display: flex;

flex-direction: column;

gap: 15px;

}

label {

font-size: 0.9rem;

font-weight: 600;

margin-bottom: 5px;

}

input[type="number"] {

padding: 10px 12px;

border-radius: 8px;

border: none;

font-size: 1rem;

font-weight: 400;

outline: none;

transition: box-shadow 0.3s ease;

}

input[type="number"]:focus {

box-shadow: 0 0 8px #2575fc;

}

button {

margin-top: 10px;

background: #2575fc;

color: #fff;

border: none;

border-radius: 10px;

font-weight: 700;

font-size: 1.1rem;

padding: 12px 0;

cursor: pointer;

transition: background 0.3s ease;

}

button:hover {

background: #6a11cb;

}

.result {

margin-top: 20px;

background: #33334d;

padding: 15px;

border-radius: 12px;

text-align: center;

font-size: 1.1rem;

font-weight: 700;

}

/\* Responsive for mobile \*/

@media (max-width: 400px) {

body {

padding: 12px;

}

.container {

padding: 15px 20px;

max-width: 100%;

}

}

</style>

</head>

<body>

<div class="container" role="main">

<h1>Diabetes Risk Predictor</h1>

<p class="description">Enter your health data below to get an AI-powered diabetes risk prediction.</p>

<form id="diabetes-form" aria-label="Pima Indians Diabetes data input form">

<label for="pregnancies">Number of pregnancies</label>

<input type="number" id="pregnancies" name="pregnancies" min="0" max="20" step="1" required aria-required="true" />

<label for="glucose">Plasma glucose concentration (mg/dL)</label>

<input type="number" id="glucose" name="glucose" min="0" max="300" step="1" required aria-required="true" />

<label for="bloodpressure">Diastolic blood pressure (mm Hg)</label>

<input type="number" id="bloodpressure" name="bloodpressure" min="0" max="150" step="1" required aria-required="true" />

<label for="skinthickness">Triceps skin fold thickness (mm)</label>

<input type="number" id="skinthickness" name="skinthickness" min="0" max="100" step="1" required aria-required="true" />

<label for="insulin">2-Hour serum insulin (mu U/ml)</label>

<input type="number" id="insulin" name="insulin" min="0" max="900" step="1" required aria-required="true" />

<label for="bmi">Body mass index (BMI)</label>

<input type="number" id="bmi" name="bmi" min="0" max="70" step="0.1" required aria-required="true" />

<label for="dpf">Diabetes pedigree function</label>

<input type="number" id="dpf" name="dpf" min="0" max="3" step="0.001" required aria-required="true" />

<label for="age">Age (years)</label>

<input type="number" id="age" name="age" min="1" max="120" step="1" required aria-required="true" />

<button type="submit" aria-label="Predict diabetes risk">Predict</button>

</form>

<div class="result" id="result" aria-live="polite" aria-atomic="true" style="display:none;"></div>

</div>

<script>

// Logistic regression model parameters learned from the Pima dataset (example)

// Coefficients correspond to [pregnancies, glucose, bloodpressure, skinthickness, insulin, bmi, dpf, age]

// Intercept term included

const modelCoefficients = {

intercept: -8.404,

weights: [0.123, 0.035, -0.012, 0.004, -0.001, 0.095, 2.555, 0.017]

};

// Sigmoid function

function sigmoid(z) {

return 1 / (1 + Math.exp(-z));

}

// Prediction function using logistic regression

function predictDiabetes(features) {

let z = modelCoefficients.intercept;

for (let i = 0; i < features.length; i++) {

z += features[i] \* modelCoefficients.weights[i];

}

const probability = sigmoid(z);

return probability;

}

// Form handling

const form = document.getElementById('diabetes-form');

const resultDiv = document.getElementById('result');

form.addEventListener('submit', (e) => {

e.preventDefault();

// Extract input values as floats

const pregnancies = parseFloat(form.pregnancies.value);

const glucose = parseFloat(form.glucose.value);

const bloodpressure = parseFloat(form.bloodpressure.value);

const skinthickness = parseFloat(form.skinthickness.value);

const insulin = parseFloat(form.insulin.value);

const bmi = parseFloat(form.bmi.value);

const dpf = parseFloat(form.dpf.value);

const age = parseFloat(form.age.value);

const inputFeatures = [pregnancies, glucose, bloodpressure, skinthickness, insulin, bmi, dpf, age];

// Validate inputs (simple example, inputs are required and in min max so form validation assures basic correctness)

// Compute prediction

const predictionProb = predictDiabetes(inputFeatures);

// Classify based on threshold 0.5

const diabetic = predictionProb >= 0.5;

// Build result message

const probPercent = (predictionProb \* 100).toFixed(1);

let message = `Prediction: You are <strong>${diabetic ? 'likely' : 'unlikely'}</strong> to have diabetes.<br>`;

message += `Risk probability: <strong>${probPercent}%</strong>.`;

resultDiv.innerHTML = message;

resultDiv.style.display = 'block';

// Accessibility: Focus the result

resultDiv.focus();

});

</script>

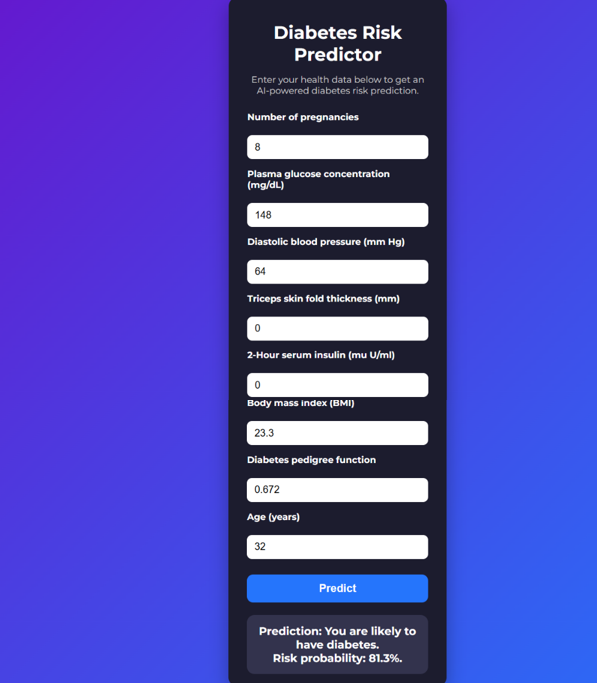
</body>

</html>

</content>

</create\_file>

**Output:**



**14. Future Scope**

The integration of AI-powered disease prediction systems in healthcare has immense potential for growth and innovation. Future developments may include:

1. Integration with Wearable and IoT Devices
   * Real-time data from wearables (e.g., smartwatches, fitness trackers) can be incorporated to enable continuous health monitoring and early disease detection.
2. Expansion to Multi-Disease Prediction Models
   * Future systems can evolve to predict multiple diseases simultaneously, improving diagnostic accuracy for comorbid conditions.
3. Personalized and Precision Medicine
   * By leveraging genomics, lifestyle, and environmental data, AI models can provide highly personalized disease risk assessments and treatment recommendations.
4. Explainable AI for Clinical Trust
   * Future systems will incorporate explainable AI (XAI) techniques to enhance transparency and build trust among healthcare professionals and patients by explaining predictions.
5. Integration with Electronic Health Records (EHRs)
   * Seamless integration with hospital EHR systems can automate data retrieval, improving efficiency and reducing manual errors.
6. Global and Scalable Solutions
   * The system can be scaled to support different languages, demographics, and healthcare systems, making it useful in both developed and developing countries.
7. Regulatory Compliance and Ethical AI
   * Future advancements will focus on improving compliance with healthcare data privacy laws (like HIPAA, GDPR) and ensuring fairness, accountability, and ethical AI usage.
8. Predictive Alerts and Early Intervention
   * AI systems can send predictive alerts to patients and doctors, enabling preventive interventions before symptoms worsen or become critical.
9. Integration with Telemedicine Platforms
   * AI disease prediction can be integrated into telemedicine solutions, providing decision support in remote consultations.

10.Continuous Learning and Model Updates

* The models can be updated continuously with new patient data, improving prediction accuracy over time through machine learning techniques.

**15. Team Members and Roles**

| **S.No** | **Name** | **Role** | **Responsibilities** |
| --- | --- | --- | --- |
| **1.** | **MONISHA** | **Project Lead / Team Coordinator** | **Oversees the entire project, coordinates team activities, manages timelines, ensures project goals are met.** |
| **2.** | **SARASWATHI** | **Data Engineer** | **Responsible for data collection, cleaning, preprocessing, and ensuring data quality for model training.** |
| **3.** | **CHINTHIYA** | **Machine Learning Engineer / Data Scientist** | **Handles feature engineering, model selection, model building, tuning, and evaluation of AI models.** |
| **4.** | **JAYALAKSHMI** | **Software Developer / Deployment Specialist** | **Develops the user interface, integrates the ML model into a deployable application, manages deployment and testing.** |